Meta Automatic Curriculum Learning

Rémy Portelas, Clément Romac, Katja Hofmann, Pierre-Yves Oudeyer

Meta Automatic Curriculum Learning

Rémy Portelas INRIA France remy.portelas@inria.fr
Clément Romac INRIA France
Katja Hofmann Microsoft Research Cambridge
Pierre-Yves Oudeyer INRIA France

Abstract: A major challenge in the Deep RL (DRL) community is to train agents able to generalize their control policy over situations never seen in training. Training on diverse tasks has been identified as a key ingredient for good generalization, which pushed researchers towards using rich procedural task generation systems controlled through complex continuous parameter spaces. In such complex task spaces, it is essential to rely on some form of Automatic Curriculum Learning (ACL) to adapt the task sampling distribution to a given learning agent, instead of randomly sampling tasks, as many could end up being either trivial or unfeasible. Since it is hard to get prior knowledge on such task spaces, many ACL algorithms explore the task space to detect progress niches over time, a costly tabula-rasa process that needs to be performed for each new learning agents, although they might have similarities in their capabilities profiles. To address this limitation, we introduce the concept of Meta-ACL, and formalize it in the context of black-box RL learners, i.e. algorithms seeking to generalize curriculum generation to an (unknown) distribution of learners. In this work, we present AGAIN, a first instantiation of Meta-ACL, and showcase its benefits for curriculum generation over classical ACL in multiple simulated environments including procedurally generated parkour environments with learners of varying morphologies. Videos and code are available at https://sites.google.com/view/meta-acl.

Keywords: Automatic Curriculum Learning, Deep Reinforcement Learning

1 Introduction

The idea of organizing the learning sequence of a machine is an old concept that stems from multiple works in reinforcement learning [1, 2], developmental robotics [3] and supervised learning [4, 5], from which the Deep RL community borrowed the term *Curriculum Learning*. Automatic CL [6] refers to *teacher* algorithms able to autonomously adapt their task sampling distribution to their evolving *student*. In DRL, ACL has been leveraged to scaffold learners in a variety of multi-task control problems, including video-games [7, 8, 9], multi-goal robotic arm manipulation [10, 11, 12, 13] and navigation in sets of environments [14, 15, 16, 17, 18]. Concurrently, multiple authors demonstrated the benefits of Procedural Content Generation (PCG) as a tool to create rich task spaces to train generalist agents [19, 20, 21]. The current limit of ACL is that, when applied to such large continuous task spaces, that often have few learnable subspaces, it either relies on 1) human expert knowledge that is hard/costly to provide (and which undermines how automatic the ACL approach is), or 2) it loses a lot of time finding tasks of appropriate difficulty through *task exploration*.

Given the aforementioned impressive results on training DRL learners with ACL to generalize over tasks (which extended the classical single-task scenarios [22, 23, 24] to multi-tasks), we propose to go further and work on training (unknown) distributions of students on continuous task spaces, thereafter referred to as *Classroom Teaching* (CT). CT defines a family of problems in which a teacher algorithm is tasked to sequentially generate multiple curricula tailored for each of its students, all having potentially varying abilities. CT differs from the problems studied in population-based developmental robotics [25] and evolutionary algorithms [26] as in CT there is no direct control over the characteristics of learners, and the objective is to foster maximal learning progress over all learners rather than iteratively populating a pool of high-performing task-expert policies. Studying CT scenarios brings DRL closer to assisted education research problems and might stimulate the design of methods that alleviate the expensive use of expert knowledge in current state of the art methods [27, 28]. CT can also be transposed to (multi-task) robotic training scenarios, e.g. when performing

iterative design improvements on a robot, which requires to train a sequence of morphologically related (yet different) robots.

Given multiple students to train, no expert knowledge, and assuming at least partial similarities between each students' optimal curriculum, current *tabula-rasa* exploratory-ACL approaches that do not reuse knowledge between different students do not seem like the optimal choice. This motivates the research of what we propose to call Meta Automatic Curriculum Learning mechanisms, that is algorithms learning to generalize ACL over multiple students. In this work we formalize this novel setup and propose a first instantiation of a Meta-ACL approach based on a former ACL algorithm [15]. Given a new student to train, our approach is based on the extraction of adapted curriculum priors from a history of previously trained students. The prior selection is performed by matching competence vectors that are built for each student through pre-testing. We show that this simple method can bring significant performance improvements over classical ACL in both a toy environment without DRL students and on Box2D parkour environments with DRL learners.

Related Work. To approach the problem of curriculum generation for DRL agents, recent works proposed multiple ACL algorithms based on the optimization of surrogate objectives such as learning progress [15, 14, 29, 11], diversity [30, 31, 32] or intermediate difficulty [17, 33, 34, 35, 16]. All these works tackled student training through independent ACL runs, while we propose to investigate how one can share information accross multiple trainings. Within DRL, *Policy Distillation* [36, 37] consists in leveraging one or several previously trained policies to perform *behavior cloning* on a new policy (e.g. to speed up training and/or to leverage task-experts to train a multi-task policy). Our work can be seen as proposing a complementary toolbox aiming to perform *Curriculum Distillation* on a continuous space of tasks.

Similar ideas were developed for supervised learning by [38, 39, 40]. In [38], authors propose an approach to infer a curriculum from past training for an image classification task: they train their network once without curriculum and use its predictive confidence for each image as a difficulty measure exploited to derive an appropriate curriculum to re-train the network. Although we are mainly interested in training a classroom of diverse students, section 4.3 presents similar experiments in a DRL scenario, showing that our Meta-ACL procedure can be beneficial for a single learner that we train once and re-train using curriculum priors inferred from the first run.

Parallel to this work, Turcheta et. al. [41] studied how to infer safety constraints (i.e. curriculum) over multiple DRL students in a data-driven way to better perform on a given single task. In their work, students are only varying by their network's initialization and their teacher assumes the existence of a pre-defined discrete set of safety constraints to choose from. By contrast, we consider the problem of training generalist students with varying morphologies (and networks initializations), with a teacher algorithm choosing tasks from a continuous task space.

Main Contributions.

- Presentation and invitation towards the study of Meta-ACL, i.e. algorithms that generalize curriculum generation in Classroom Teaching scenarios. Formalization of the interaction flows between Meta-ACL algorithms and (unknown) student distributions.
- Presentation and analysis of AGAIN, a first algorithmic instantiation of Meta-ACL which leverages ALP-GMM, a recent ACL algorithm [15].
- Design of a toy-environment and of a parametric Box2D Parkour environment featuring a multi-modal distribution of possible agent embodiments well suited to study Meta-ACL.

2 Meta Automatic Curriculum Learning Framework

Black-box students The Meta-ACL framework assumes the existence of policy learners, a.k.a students, capable of interacting in episodic control tasks. Their optimization objective is the maximization of some performance measure P w.r.t the task (e.g. episodic reward, exploration). To make the framework problem-independant, we do not assume expert knowledge over the task space w.r.t the student distribution, e.g. task subspaces could be trivial for some students and unfeasible for others. The objective of Meta-ACL is precisely to autonomously infer such prior knowledge from experience in scenarios where human expert knowledge is either hard or impossible to use.

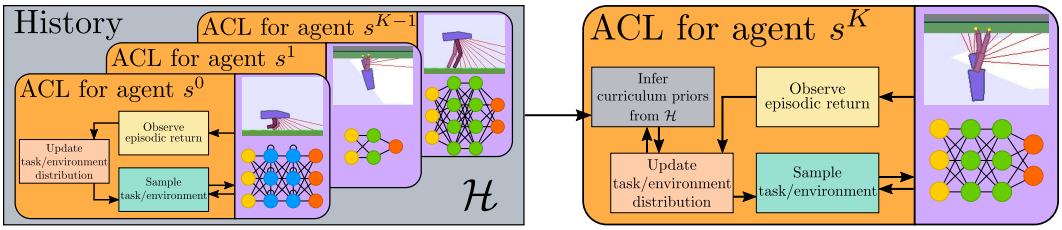


Figure 1: In *Meta Automatic Curriculum Learning* (Meta-ACL), the objective is to leverage previous teaching experience to improve the curriculum generation of a new agent, whose embodiment and learning mechanisms have potentially never been seen before: the teacher has to *generalize* over students.

Similarly, we consider a black-box teaching scenario, i.e. we do not assume the knowledge of which learning mechanisms are used by students (e.g. DRL policies, evolutionary algorithms, ...).

Automatic Curriculum Learning Given a continuous *task space* $\mathcal{A}$, a single black-box student s (assumed non-resettable, as in classical ACL scenarios), and a budget of E episodes, one can frame the ACL optimization objective as:

$$\max \int_{a \sim \mathcal{A}} P^E_{s,a} \, \mathrm{d}a, \tag{1}$$

with $P^E_{s,a}$ the end performance of student s on task a (e.g. exploration score, cumulative reward). Since direct optimization of such a post-training performance is difficult, ACL is often approached using proxy objectives (e.g. intermediate difficulty). Given one such proxy objective, an ACL algorithm usually relies on observing episodic behavioral features of its student w.r.t to proposed tasks (e.g. episodic rewards), allowing to infer a competence status $o \in \mathcal{O}$ of its learner (e.g. progress regions in $\mathcal{A}$) which conditions task sampling. This sequential task selection unrolled by ACL methods along a student's training can be transposed into a policy search problem on a high-level non-episodic POMDP. While interacting within this POMDP, an ACL policy $\Pi(o, s) \rightarrow \rho(\mathcal{A})$ proposes task distributions $\rho(\mathcal{A}) \subset \mathcal{A}$ to its student, observes the resulting behavioral features to update its competence status o, and collects objective-dependant rewards (e.g. learning progress).

In practice, approaching this task-level control problem with classical DRL algorithms is challenging because of sample efficiency: an ACL policy has to be learned and exploited along interaction windows typically around a few tens of thousands of steps. This has to be compared to the tens of millions or sometimes billions of interaction steps necessary to train a DRL policy for robotic control tasks. For this reason, most recent ACL research has focused on reducing the teaching problem into a Multi Armed Bandit setup, which ignores the sequential dependency over student states implied in POMDP settings [14, 29, 11, 15]. Although out of the scope of this paper, the use of classical DRL approaches as Meta-ACL algorithms is worth investigating in future work.

Meta-ACL for Classroom Teaching. We now present the concept of Meta-ACL applied to a Classroom Teaching scenario, i.e. there is no longer a single student s to be trained, but a set of students with varying abilities (e.g. due to morphology and/or learning mechanisms) sequentially drawn from an unknown distribution $\mathcal{S}$. The notion of meta-learning refers to *any type of learning guided by prior experience with other tasks* [42]. In meta-RL, agents are *learning to learn* to act [43], i.e. their objective is to maximize performance on previously unseen test tasks after 0 or a few learning updates. In other words, it is about leveraging knowledge from previously encountered tasks to generalize on new tasks. We propose to extend this concept into Meta-ACL, that is, algorithms that are *learning to learn* to teach, i.e. they leverage knowledge from curricula built for previous students to improve the curriculum generation for new ones. More precisely, a Meta-ACL algorithm can be formulated as a function:

$$\begin{aligned} & f(\Pi, \mathcal{H}, s^K) \rightarrow \Pi^{'} \;\; s.t. \;\; \mathcal{H} = [\tau_{s^0}, \tau_{s^1}, ..., \tau_{s^{K-1}}] \\ \text{and} \quad & \tau_{s^i} = [(\rho^0(\mathcal{A}), o^0), (\rho^1(\mathcal{A}), o^1), ..., (\rho^T(\mathcal{A}), o^T)], \end{aligned} \tag{2}$$

with $\mathcal{H}$ the history of past K training trajectories τ_s resulting from the scaffolding of K previous students with ACL policy Π, and s^K the current student. Given our formalization of ACL (see eq.

1), the Meta-ACL optimization objective can be expressed as follows:

$$\max_{f} \int_{s\sim\mathcal{S}} \int_{a\sim\mathcal{A}} P^{E}_{s,a} \, \mathrm{d}a \, \mathrm{d}s. \tag{3}$$

As in the case of the ACL optimization objective expressed in eq. 1, direct optimization of eq. 3 is difficult as it implies the joint maximization of multiple students' performance. In our experiments, we reduce the Meta-ACL optimization objective to the sequential independent optimization of a set of sampled students (with the hope to maximize performance over the entire set). Figure 1 provides a visual transcription of the workflow of a Meta-ACL algorithm. While in our experiments we use a fixed-size history $\mathcal{H}$ of ACL-trained students (to make our experiments computationally tractable), $\mathcal{H}$ could be grown incrementally by collecting training trajectories online from Meta-ACL trainings.

3 A first Meta-ACL approach: Alp-Gmm And Inferred Progress Niches

In this section, we present our proposed Meta-ACL algorithm within the formalism described in section 2. We assume a history $\mathcal{H}$ resulting from the training of a classroom of K students with ACL, and the objective is to leverage this knowledge to better train a set of new students. For its simplicity and its non-reliance on expert knowledge, we use ALP-GMM [15] as our underlying ACL teacher. Given this, after the initial training of K students with ALP-GMM, our approach consists of two main algorithmic components: 1) How to select useful curriculum priors from $\mathcal{H}$ w.r.t. a new learner, and 2) how to use these priors to improve the curriculum generation w.r.t. this new learner?

ALP-GMM ALP-GMM [15] is a Learning Progress (LP) based ACL technique for continuous task spaces that does not assume prior knowledge. ALP-GMM frames the task sampling problem into a non-stationary Multi-Armed bandit setup [44] in which arms are Gaussians spanning over the task space. The utility of each Gaussian is defined with a local LP measure derived from episodic reward comparisons. The essence of ALP-GMM is to periodically fit a Gaussian Mixture Model (GMM) on recently sampled tasks' parameters *concatenated with their respective LP*. This periodically updated GMM used for task sampling can be seen as the evolving competence status o described in section 2. The Gaussian from which to sample a new task is chosen proportionally to its mean LP dimension. Task exploration happens initially through a bootstrapping period of random task sampling and during training through residual random task sampling.

Knowledge Component based prior selection. For a new student $s^K \sim \mathcal{S}$, how to infer adapted priors over its capabilities (w.r.t to a given task space) from the history $\mathcal{H}$ of previously trained students? This problem is closely related to knowledge assessment in Intelligent Tutoring Systems setups studied in the educational data mining literature [45, 46]. Inspired by these works, we propose to use student pre-tests to derive a Knowledge Component (KC) vector $KC^{pre} \in \mathbb{R}^m$ for all trained students. Each dimensions of KC^{pre} contains the episodic return of the student on the corresponding pre-test task. Given that we do not assume access to expert knowledge, we build this pre-test task set by selecting m tasks uniformly over the task space. We use the same task set to build a post-training KC vector $KC^{post} \in \mathbb{R}^m$ whose dimensions are summed up to get a score $j_s \in \mathbb{R}$, used to evaluate the end performance of students in $\mathcal{H}$. Given a new student s^K, we can now 1) pre-train it (using ALP-GMM), then 2) pre-test it to get its KC vector $KC^{pre}_{s^K}$, and 3) infer the k most similar previously trained students in KC space (using a k-nearest neighbor algorithm) and extract their respective training trajectory τ_s from $\mathcal{H}$. We then use the training trajectory of the student with maximal post-training score j_s among those k.

Inferred progress Niches (IN). Given that the KC-based student selection identified the training trajectory τ_{s^i} as the most promising for s^K, and assuming ALP-GMM as the underlying ACL algorithm used for s^i, we can derive an expert curriculum from τ_{s^i} by first considering the sequence of GMMs $\mathcal{C}_{raw}$ that were periodically fitted along training:

$$\mathcal{C}_{raw} = \{p(1), ..., p(T)\} \ , \ p(t) = \sum_{i=1} LP_{ti} \mathcal{N}(\boldsymbol{\mu_{ti}}, \boldsymbol{\Sigma_{ti}}), \tag{4}$$

with T the total number of GMMs in the list and LP_{ti} the Learning Progress of the i^{th} Gaussian from the t^{th} GMM. By keeping only Gaussians with LP_{ti} above a predefined threshold δ_{LP}, we can

get a curated list $\mathcal{C}$ containing only Gaussians located on task subspaces on which s^i experienced learning progress. We name the resulting meta-learned expert curriculum approach Infered progress Niches (IN) and propose 3 variants to select which GMM from $\mathcal{C}$ is used to sample tasks over time along the training of s^K:

- **Pool-based (IN-P)** A rather crude approach is to disregard the ordering of $\mathcal{C}$ and merge the trajectory of GMMs into a single large GMM.
- **Time-based (IN-T)** In this version $\mathcal{C}$ is stepped in periodically at the same rate than the preliminary ALP-GMM run (ie. a new GMM every 250 episodes in our experiments).
- **Reward-based (IN-R)** A more adaptive option is to iterate over $\mathcal{C}$ only once the mean episodic reward over tasks recently sampled from the current GMM matches or surpasses the mean episodic reward recorded during the initial ALP-GMM run (on the same GMM).

Regardless of the selection process, given a GMM, a new task is selected by sampling its parameters from a Gaussian selected proportionally to its LP_{ti} value.

Our proposed approach: AGAIN. Simply using one of the 3 proposed IN algorithms directly for s^K lacks adaptive mechanisms towards the characteristics of the new student (e.g. new embodiment, different initial parameters, ...), which could lead to failure cases where the expert curriculum misses important aspects of training (e.g. detecting task subspaces that are being forgotten). Additionally, for the IN-R variant, which schedules when to iterate over the expert curriculum, the meta-learned ACL algorithm must have the capacity to emancipate from the expert curriculum once the trajectory is completed. As such we propose to combine IN with an ALP-GMM teacher (with low task-exploration). The resulting Alp-Gmm And Inferred progress Niches approach (AGAIN) samples tasks from a GMM that is composed of the current mixture of both ALP-GMM and IN. See fig. 2 for a schematic pipeline and appendix A & B for implementation details and pseudo-code algorithms.

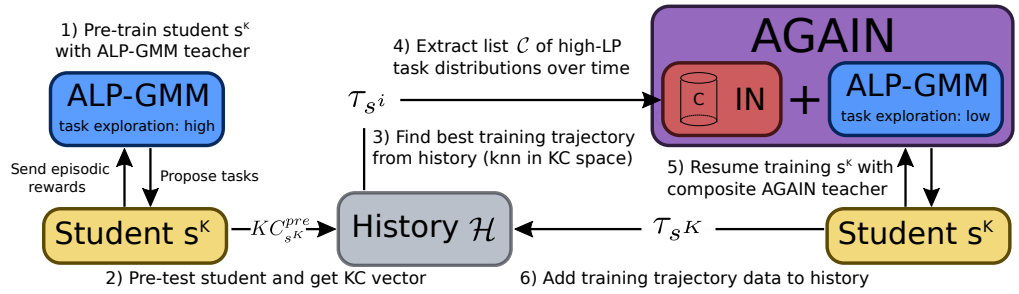


Figure 2: Schematic pipeline of Alp-Gmm And Inferred progress Niches (AGAIN), our proposed approach, which first leverages a preliminary run with a high-exploration ALP-GMM curriculum generator, and uses student pre-testing to infer an expert curriculum combined with a low-exploration ALP-GMM.

4 Experiments and Results

We organize the analysis of our proposed Meta-ACL algorithm around 3 experimental questions:

- What are the properties and important components of AGAIN? In this section we will leverage a toy environment without DRL students to conduct systematic experiments.
- Does AGAIN scale well to Meta-ACL scenarios with DRL students? Here we will present a new Parkour environment that will be used to conduct our experiments.
- Can AGAIN be used for single learners? Here we will show that it can be useful to derive curriculum priors even for a single student (i.e. without any student History $\mathcal{H}$).

Considered baselines and AGAIN variants. In the following experiments we compare our 3 proposed ways of deriving an expert curriculum from a selected past training trajectory (see sec. 3), giving the *AGAIN-R*, *AGAIN-T* and *AGAIN-P* conditions. We also consider variants where 1) we directly use the expert curriculum instead of combining it with ALP-GMM (*IN-R*, *IN-T* and *IN-P* conditions), and 2) where we select a training trajectory at random (*AGAIN_RND*) or using the

ground truth student distribution (*AGAIN_GT*). We compare these Meta-ACL variants to ACL approaches such as random curriculum generation (*Random*), *ALP-GMM* and either Adaptive Domain Randomization (*ADR*) [34] or an expert-made *Oracle* curriculum. See appendix C for details.

4.1 Analysing Meta-ACL in a toy environment

To provide in-depth experiments on AGAIN, we first emancipate from DRL students through the use of a modified version of the toy testbed presented in [15]. The objective of this environment is to simulate the learning of a student within a 2D parameter space $\mathcal{P} = [0, 1]^2$. The parameter space is uniformly divided in 400 square cells $C \subset \mathcal{P}$, and each parameter $p \in \mathcal{P}$ sampled by the teacher is directly mapped to an episodic reward r_p based on sampling history and whether C is considered "locked" or "unlocked". Three rules enforce reward collection in $\mathcal{P}$: 1) Every cell C starts "locked", except a randomly chosen one that is "unlocked". 2) If C is "unlocked" and $p \in C$, then $r_p = min(|C|, 100)$, with $|C|$ the cumulative number of parameters sampled within C while being "unlocked" (if C is "locked", then $r_p = 0$). Finally, 3) If $|C| >= 75$, adjacent cells become "unlocked". Given these rules, one can model students with different curriculum needs by assigning them different initially unlocked cells, which itself models what is "easy to learn" initially for a given student, and from where it can expand.

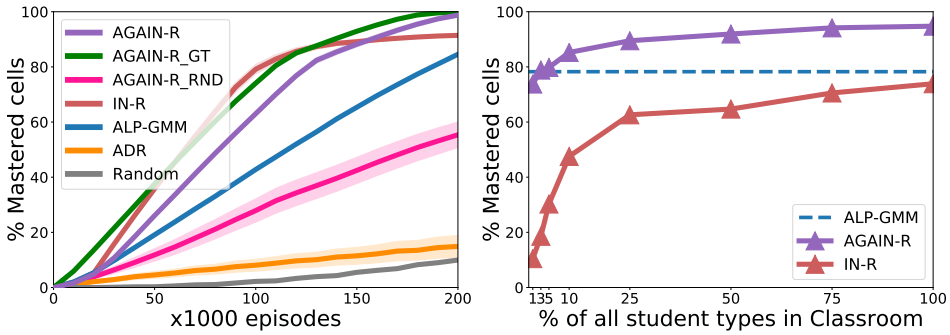


Figure 3: **left:** By leveraging meta-learned curriculum priors w.r.t to its students, AGAIN-R outperforms regular ACL approaches. Avg. perfs. with *sem* (standard error of the mean) plotted, 48 seeds. **right:** Impact of classroom size and sparsity on Meta-ACL performances. Post-training (200k ep.) avg perfs. plotted, 96 seeds.

Results Instead of performing a pre-test to construct the KC vector of a student, we directly compute it by concatenating $|C|$ for all cells, giving a 400-dimensional KC vector. This vector is computed after 20k training episodes out of 200k. We thereafter only report results for AGAIN-R, the highest performing AGAIN variant w.r.t. AGAIN-T and AGAIN-P (as shown in App. D). To study AGAIN, we first populate our Training trajectory history $\mathcal{H}$ by training with ALP-GMM an initial classroom of 128 students drawn randomly from 4 fixed possible student types (i.e. 4 possible initially unlocked cell positions), and then test it on a new fixed set of 48 random students.

Comparative analysis - Figure 3 (left) showcases performance across training for our considered Meta-ACL conditions and ACL baselines. Both AGAIN-R and IN-R significantly outperform ALP-GMM ($p < .001$ for both, using Welch's t-test at 200k episodes). The initial performance advantage of IN-R w.r.t AGAIN-R is due to the greedy nature of IN-R, which only exploits the expert curriculum while AGAIN complements it with ALP-GMM for exploration. By the end of training, AGAIN-R outperforms IN-R ($p < .001$) thanks to its ability to emancipate from the curriculum priors it initially leverages. The regular KC-based curriculum priors selection used in AGAIN-R outperformed the random selection used in AGAIN-R_RND ($p < .001$ at 200k episodes), while being not significantly inferior to the Ground Truth variant AGAIN-R_GT ($p = 0.16$). Because we assume no expert knowledge over the set of students to train, i.e. their respective initial learning subspace is unknown, ADR – which relies on being given an initial easy task – fails to train most students when given randomly selected starting subspace (among the 4 possible ones). By contrast, this showcases the ability of AGAIN to autonomously and efficiently infer such expert knowledge.

Varying classroom size experiment - An important property that must be met by a meta-learning procedure is to have a monotonic increase of performance as the database of information being

leveraged increases. Another important expected aspect of Meta-ACL is whether the approach is able to generalize to students that were never seen before. To assess whether these properties hold on AGAIN, we consider the full student distribution of the toy environment, i.e. 400 possible student types. We populate a new history $\mathcal{H}$ by training (with ALP-GMM) a 400-students classroom (one per student type). We then analyse the end performance of AGAIN-R and IN-R on a fixed test set of 96 random students when given increasingly smaller subsets of $\mathcal{H}$. The smaller the subset, the harder it becomes to generalize over new students. Results, shown in fig. 3 (right), demonstrate that both AGAIN and IN do have monotonic performance increasements as the classroom grows. With as little as 10% of possible students in the classroom, AGAIN statistically significantly ($p < .001$) outperforms ALP-GMM on the new student set, i.e. it generalizes to never seen before students.

4.2 Meta-ACL for DRL students in the Parkour environment

To study Meta-ACL with DRL students, we present a Box2D Parkour environment with a 2D parametric PCG that encodes a large space of tasks (see fig. 4). The first parameter controls the spacing between walls that are positioned along the track, while the second parameter sets the y-position of a gate that is added to each wall. Positive rewards are collected by going forward. To simulate a multi-modal distribution of students well suited to study Meta-ACL, we randomize the student's morphology for each new training (i.e. each seed): It can be embodied in either a bipedal walker, which will be prone to learn tasks with near-ground gate positions, or a two-armed climber, for which tasks with near-roof gate positions are easiest. We also randomize the student's limb sizes which can vary from the length visible in fig. 4 to 50% shorter.

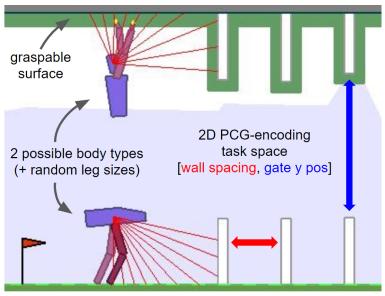


Figure 4: Our proposed parametric Parkour env. to study Meta-ACL with DRL students.

Results In the following experiments our Meta-ACL variants leverage a history $\mathcal{H}$ built from a classroom of 128 randomly drawn Soft-Actor-Critic [47] students (i.e. varying embodiments and initial policy weights) trained with ALP-GMM. We then compare ACL and Meta-ACL variants on a fixed set of 64 new students and report the mean percentage of mastered environments (i.e. $r > 230$) from 2 fixed expert test sets (one per embodiment type) across training. The KC vector is built using a uniform pre-test set of $m = 225$ tasks, performed after 2 millions agent steps out of 10. See appendix E for additional experimental details.

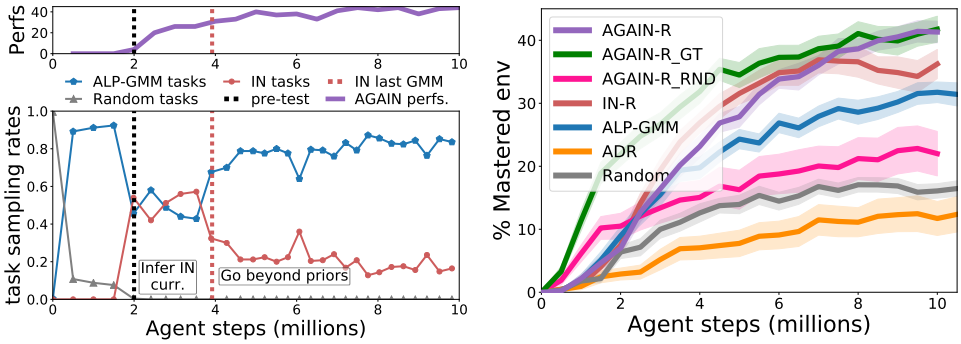


Figure 5: **left:** Example of evolution of task sampling when using AGAIN-R in the Parkour env. (1 seed). **right:** Average performances of AGAIN with variants and baselines in the Parkour env.. 64 seeds, sem plotted.

Qualitative view - Figure 5 (left) showcases the evolution of task sampling when using AGAIN to train a new student. Three distinct phases emerge: 1) A pre-training exploratory phase used to gather information about the student's capabilities, 2) After building the KC vector and inferring the most appropriate curriculum priors from $\mathcal{H}$, AGAIN paces through the resulting IN curriculum while mixing it to ALP-GMM, and 3) AGAIN emancipates from IN after completing it.

Comparative analysis - As shown in figure 5 (right), through its use of curriculum priors, AGAIN-R outperforms ALP-GMM on Parkour, mastering an average of 41% of the test set at 10M steps,

compared to 31% for ALP-GMM ($p < .001$) after 10.5M steps (0.5M training steps added to account for AGAIN-R additional pre-test time). AGAIN-R performs better than its AGAIN-R_RND random prior selection variant, and is not statistically different ($p = 0.8$) from ground truth sampling (AGAIN-R_GT), although only by the end of training. While AGAIN-R and IN-R initially have comparable performances, after 7 Millions training steps, – a point at which most students trained with IN-R or AGAIN-R reached the last IN GMM –, AGAIN-R outperforms IN-R by the end of training ($p < 0.02$). This showcases the advantage of emancipating from the expert curriculum once completed. As in the toy environment experiments, when given randomly selected starting subspaces (since we assume no expert knowledge), ADR fails to train most students.

4.3 Applying Meta-ACL to a single student: Trying AGAIN instead of trying longer

Given a single DRL student to train (i.e. no history $\mathcal{H}$), and if we do not assume access to expert knowledge, current ACL approaches leverage task-exploration (as in ALP-GMM). We hypothesize that these additional tasks presented to the DRL learner have a cluttering effect on the gathered training data, which adds noise in its already brittle gradient-based optimization and leads to sub-optimal performances. We propose to address this problem by modifying AGAIN to fit this no-history setup and by allowing to restart the student along training. More precisely, instead of pre-testing the student to find appropriate curriculum priors in $\mathcal{H}$, we split the training of the target student into a two stage approach where 1) the DRL student is first trained with ALP-GMM (with high-exploration), and then 2) we extract curriculum priors from the training trajectory of the first run and use them to re-train the same agent *from scratch*.

Results. We test our modified AGAIN along with variants and baselines on a parametric version of BipedalWalker proposed in [15], which generates walking tracks paved with stumps whose height and spacing are defined by a PCG-encoding 2-D parameter vector. As in their work, we test our approaches with both the default walker and a modified short-legged walker, which constitutes an even more challenging scenario (as the task space is unchanged). Performance is measured by tracking the percentage of mastered tasks from a fixed test set. See App. F for a complete analysis. Figure 6 showcases our proposed approach on the short walker setup (with a SAC student [47]). On this short walker scenario, mixing ALP-GMM with IN-R is essential: while IN-R end performances are not statistically significantly superior to ALP-GMM, AGAIN-R clearly outperforms ALP-GMM ($p < 0.01$), reaching a mean end performance of 19.0. The difference in end-performance between AGAIN-R and Oracle, our hand-made curriculum using privileged information who obtained 20.1, is not significant ($p = 0.6$).

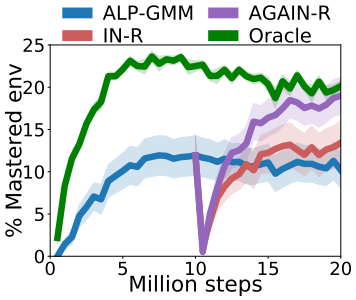


Figure 6: Given a single DRL student to train, AGAIN outperforms ALP-GMM in a parametric BipedalWalker environment. sem plotted, 32 seeds.

5 Conclusion and Discussion

In this work we attempted to motivate and formalize the study of Classroom Teaching problems, in which a set of diverse students have to be trained optimally, and we proposed to attain this goal through the use of Meta-ACL algorithms. We then presented AGAIN, a first Meta-ACL approach, and demonstrated its advantages over ACL baselines and variants for CT problems in both a toy environment and in a new parametric Parkour environment with DRL learners. We also showed how AGAIN can bring performance gains over ACL in classical single student ACL scenarios.

In future work, AGAIN could be improved by using adaptive approaches to build compact pre-test sets, e.g. using decision tree based test pruning methods, or by combining curriculum priors from multiple previously trained learners. While AGAIN is built on top of an existing ACL algorithm, developing an end-to-end Meta-ACL algorithm that generates curricula using a DRL teacher policy trained across multiple students is also a promising line of work to follow. Additionally, this work opens-up exciting new perspectives in transferring Meta-ACL methods to educational data-mining, e.g. in MOOC scenarios, given a previously trained pilot classroom, one could use Meta-ACL to infer adaptive curricula for new students.

Acknowledgments

This work was supported by Microsoft Research through its PhD Scholarship Programme.

References

[1] O. G. Selfridge, R. S. Sutton, and A. G. Barto. Training and tracking in robotics. In *IJCAI*, 1985.

[2] J. Schmidhuber. Curious model-building control systems. In *IJCNN*. IEEE, 1991.

[3] P.-Y. Oudeyer, F. Kaplan, and V. V. Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE trans. on evolutionary comp.*, 2007.

[4] J. L. Elman. Learning and development in neural networks: the importance of starting small. *Cognition*, 48(1):71 – 99, 1993. ISSN 0010-0277.

[5] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *ICML*, 2009.

[6] R. Portelas, C. Colas, L. Weng, K. Hofmann, and P.-Y. Oudeyer. Automatic curriculum learning for deep rl: A short survey. *IJCAI*, 2020.

[7] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *CVPR*, 2017.

[8] Y. Burda, H. Edwards, A. J. Storkey, and O. Klimov. Exploration by random network distillation. *ICLR*, 2019.

[9] T. Salimans and R. Chen. Learning montezuma's revenge from a single demonstration. *NeurIPS*, 2018.

[10] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. P. Abbeel, and W. Zaremba. Hindsight experience replay. In *NeurIPS*, 2017.

[11] C. Colas, P.-Y. Oudeyer, O. Sigaud, P. Fournier, and M. Chetouani. Curious: Intrinsically motivated modular multi-goal reinforcement learning. In *ICML*, 2019.

[12] G. Cideron, M. Seurin, F. Strub, and O. Pietquin. Self-educated language agent with hindsight experience replay for instruction following. *ViGIL, NeurIPS Workshop*, 2019.

[13] P. Fournier, O. Sigaud, M. Chetouani, and P. Oudeyer. Accuracy-based curriculum learning in deep reinforcement learning. *arXiv*, 2018.

[14] T. Matiisen, A. Oliver, T. Cohen, and J. Schulman. Teacher-student curriculum learning. *IEEE TNNLS*, 2017.

[15] R. Portelas, C. Colas, K. Hofmann, and P.-Y. Oudeyer. Teacher algorithms for curriculum learning of deep rl in continuously parameterized environments. *CoRL*, 2019.

[16] B. Mehta, M. Diaz, F. Golemo, C. J. Pal, and L. Paull. Active domain randomization. *CoRL*, 2019.

[17] C. Florensa, D. Held, X. Geng, and P. Abbeel. Automatic goal generation for reinforcement learning agents. In *ICML*, 2018.

[18] P. Klink, C. D'Eramo, J. Peters, and J. Pajarinen. Self-paced deep reinforcement learning. *arXiv*, 2020.

[19] S. Risi and J. Togelius. Procedural content generation: From automatically generating game levels to increasing generality in machine learning. *arXiv*, 2019.

[20] N. Justesen, R. R. Torrado, P. Bontrager, A. Khalifa, J. Togelius, and S. Risi. Illuminating generalization in deep reinforcement learning through procedural level generation. *NeurIPS Deep RL Workshop*, 2018.

[21] K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman. Quantifying generalization in reinforcement learning. *ICML*, abs/1812.02341, 2019.

[22] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[23] J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz. Trust region policy optimization. In *ICML*, 2015.

[24] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. In *ICLR*, 2016.

[25] S. Forestier, R. Portelas, Y. Mollard, and P. Oudeyer. Intrinsically motivated goal exploration processes with automatic curriculum learning. *arXiv*, 2017.

[26] R. Wang, J. Lehman, J. Clune, and K. O. Stanley. Paired open-ended trailblazer (POET): endlessly generating increasingly complex and diverse learning environments and their solutions. *arXiv*, 2019.

[27] B. Clément, D. Roy, P.-Y. Oudeyer, and M. Lopes. Multi-Armed Bandits for Intelligent Tutoring Systems. *Journal of Educational Data Mining (JEDM)*, 7(2):20–48, June 2015.

[28] K. R. Koedinger, E. Brunskill, R. S. J. de Baker, E. A. McLaughlin, and J. C. Stamper. New potentials for data-driven intelligent tutoring system development and optimization. *AI Magazine*, 34(3):27–41, 2013.

[29] S. Mysore, R. Platt, and K. Saenko. Reward-guided curriculum for robust reinforcement learning. *Workshop on Multi-task and Lifelong Reinforcement Learning at ICML*, 2019.

[30] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine. Diversity is all you need: Learning skills without a reward function. *arXiv*, 2018.

[31] A. Jabri, K. Hsu, A. Gupta, B. Eysenbach, S. Levine, and C. Finn. Unsupervised curricula for visual meta-reinforcement learning. In *NeurIPS*. 2019.

[32] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying count-based exploration and intrinsic motivation. In *NeurIPS*, 2016.

[33] S. Racanière, A. Lampinen, A. Santoro, D. Reichert, V. Firoiu, and T. Lillicrap. Automated curricula through setter-solver interactions. *ICLR*, 2020.

[34] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q.-M. Yuan, W. Zaremba, and L. Zhang. Solving rubik's cube with a robot hand. *ArXiv*, 2019.

[35] C. Florensa, D. Held, M. Wulfmeier, and P. Abbeel. Reverse curriculum generation for reinforcement learning. *CoRL*, 2017.

[36] Y. W. Teh, V. Bapst, W. Czarnecki, J. Quan, J. Kirkpatrick, R. Hadsell, N. M. O. Heess, and R. Pascanu. Distral: Robust multitask reinforcement learning. In *NIPS*, 2017.

[37] W. M. Czarnecki, R. Pascanu, S. Osindero, S. M. Jayakumar, G. Swirszcz, and M. Jaderberg. Distilling policy distillation. *AISTATS*, 2019.

[38] G. Hacohen and D. Weinshall. On the power of curriculum learning in training deep networks. In K. Chaudhuri and R. Salakhutdinov, editors, *ICML*, 2019.

[39] T. Furlanello, Z. C. Lipton, M. Tschannen, L. Itti, and A. Anandkumar. Born-again neural networks. In *ICML*, pages 1602–1611, 2018.

[40] J. Yim, D. Joo, J. Bae, and J. Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. *CVPR*, pages 7130–7138, 2017.

[41] M. Turchetta, A. Kolobov, S. Shah, A. Krause, and A. Agarwal. Safe reinforcement learning via curriculum induction. *ArXiv*, 2020.

[42] J. Vanschoren. Meta-learning: A survey. *arXiv*, 2018.

[43] J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick. Learning to reinforcement learn. *arXiv*, 2016.

[44] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77, 2002.

[45] J.-J. Vie, F. Popineau, É. Bruillard, and Y. Bourda. Automated Test Assembly for Handling Learner Cold-Start in Large-Scale Assessments. *IJAIED*, 2018.

[46] J.-J. Vie. *Cognitive diagnostic computerized adaptive testing models for large-scale learning*. Thesis, Université Paris Saclay (COmUE), 2016.

[47] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *ICML*, 2018.

[48] H. Bozdogan. Model selection and akaike's information criterion (aic): The general theory and its analytical extensions. *Psychometrika*, 52(3):345–370, Sep 1987.

A ALP-GMM

ALP-GMM [15] relies on an empirical per-task computation of Absolute Learning Progress (ALP), allowing to fit a GMM on a concatenated space composed of tasks' parameters and respective ALP. Given a task $a_{new} \in \mathcal{A}$ whose parameter is $p_{new} \in \mathcal{P}$ and on which the student's policy collected the episodic reward $r_{new} \in \mathbb{R}$, Its ALP is computed using the closest previous tasks a_{old} (Euclidean distance) with associated episodic reward r_{old}:

$$alp_{new} = |r_{new} - r_{old}| \tag{5}$$

All previously encountered task's parameters and their associated ALP, parameter-ALP for short, recorded in a history database H, are used for this computation. Contrastingly, the fitting of the GMM is performed every N episodes on a window $\mathcal{W}$ containing the N most recent parameter-ALP. The resulting mean ALP dimension of each Gaussian of the GMM is used for proportional sampling. To adapt the number of components of the GMM online, a batch of GMMs having from 2 to k_{max} components is fitted on $\mathcal{W}$, and the best one, according to Akaike's Information Criterion [48], is kept as the new GMM. In all of our experiments we use the same hyperparameters as in [15] ($N = 250$, $k_{max} = 10$), except for the percentage of random task sampling ρ_{rnd} which we set to 10% (we found it to perform better than 20%) when running ALP-GMM. See algorithm 1 for pseudo-code and figure 7 for a schematic pipeline. Note that in the main body of this paper we refer to ALP as LP for simplicity (ie. LP_{ti} in $\mathcal{C}$ from eq. 4 is equivalent to the mean ALP of Gaussians in ALP-GMM).

Algorithm 1 Absolute Learning Progress Gaussian Mixture Model (ALP-GMM)

Require: Student policy s_θ, parametric procedural environment generator E, bounded parameter space $\mathcal{P}$, probability of random sampling ρ_{rnd}, fitting rate N, max number of Gaussians k_{max}

Initialize s_θ
Initialize parameter-ALP First-in-First-Out window $\mathcal{W}$, set max size to N
Initialize parameter-reward history database H
loop N times ▷ Bootstrap phase
    Sample random $p \in \mathcal{P}$, send $E(a \sim \mathcal{A}(p))$ to s_θ, observe episodic reward r_p
    Compute ALP of p based on r_p and H (see equation 5)
    Store (p, r_p) pair in H, store (p, ALP_p) pair in $\mathcal{W}$
loop ▷ Stop after K inner loops
    Fit a set of GMM having 2 to k_{max} kernels on $\mathcal{W}$
    Select the GMM with best Akaike Information Criterion
    loop N times
        $\rho_{rnd}\%$ of the time, sample a random parameter $p \in \mathcal{P}$
        Else, sample p from a Gaussian chosen proportionally to its mean ALP value
        Send $E(a \sim \mathcal{A}(p))$ to student s_θ and observe episodic reward r_p
        Compute ALP of p based on r_p and H
        Store (p, r_p) pair in H, store (p, ALP_p) pair in $\mathcal{W}$
Return s_θ

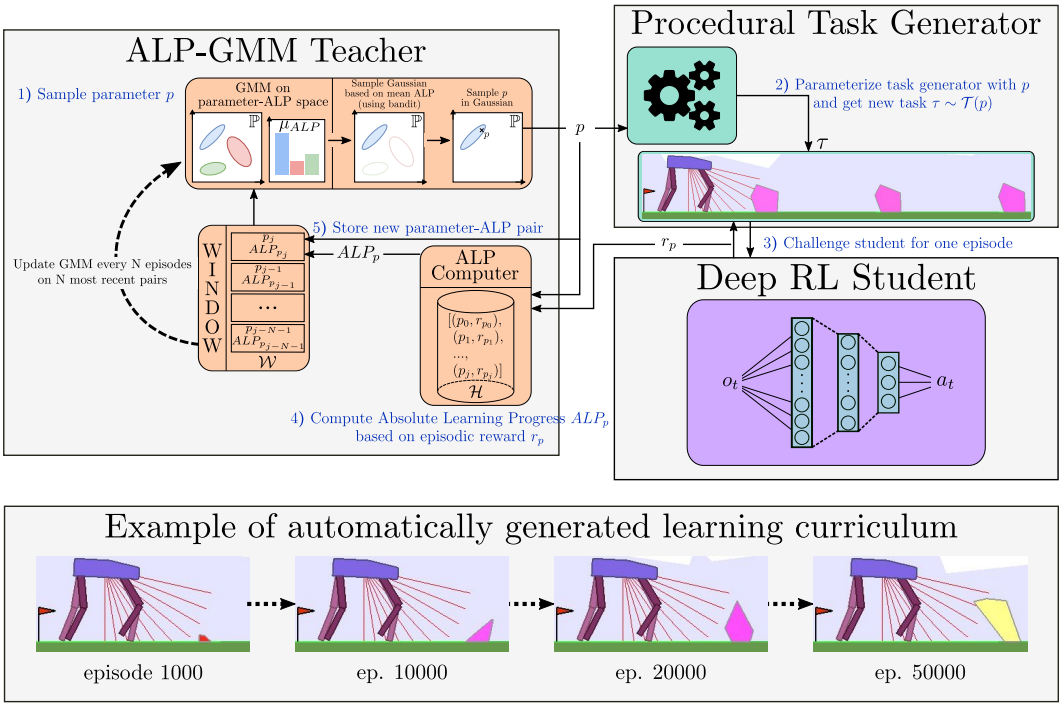


Figure 7: Schematic view of an ALP-GMM teacher's workflow from [15]

B AGAIN

IN variants. In order to filter the list $\mathcal{C}_{raw}$ (see eq. 4) of GMMs extracted from a training trajectory τ_s selected in training trajectory history $\mathcal{H}$ into $\mathcal{C}$ and use it as an expert curriculum, we remove any Gaussian with a LP_{ti} below $\delta_{LP} = 0.2$ (the LP dimension is normalized between 0 and 1, which requires to choose an approximate potential reward range, set to $[-150, 350]$ for all experiments on Box2D locomotion environments (sec. 4.2 and sec. 4.3). When all Gaussians of a GMM are discarded, the GMM is removed from $\mathcal{C}$. In practice, it allows to 1) remove non-informative GMMs corresponding to the initial exploration phase of ALP-GMM, when the learner has not made any progress (hence no LP detected by the teacher), and 2) remove an entire training trajectory τ_s if ALP-GMM never detected high-LP Gaussians, i.e. it failed to train student s. $\mathcal{C}$ is then iterated over to generate a curricula with either of the Time-based (see algo. 3), Pool-based (see algo 4) or Reward-based (see algo 5) IN. The IN-P approach does not require additional hyperparameters. The IN-T requires an update rate N to iterate over $\mathcal{C}$, which we set to 250 (same as the fitting rate of ALP-GMM). The IN-R approach requires to extract additional data from the first run, in the form of a list $\mathcal{R}_{raw}$:

$$\mathcal{R}_{raw} = \{\mu_r^1, ..., \mu_r^t, \mu_r^T\} \;\; s.t \;\; |\mathcal{R}_{raw}| = |\mathcal{C}_{raw}|, \tag{6}$$

with T the total number of GMMs in the first run (same as in $\mathcal{C}_{raw}$), and μ_r^t the mean episodic reward obtained by the first DRL agent during the last 50 tasks sampled from the t^{th} GMM. $\mathcal{R}$ is simply obtained by removing any μ_r^t that corresponds to a GMM discarded while extracting $\mathcal{C}$ from $\mathcal{C}_{raw}$. The remaining rewards are then used as thresholds in IN-R to decide when to switch to the next GMM in $\mathcal{C}$.

AGAIN In AGAIN (see algo. 6), the idea is to use both IN (R,T or P) and ALP-GMM (without the random bootstrapping period) for curriculum generation. We combine the changing GMM of IN and ALP-GMM over time, simply by building a GMM G containing Gaussians from the current GMM of IN and ALP-GMM. By selecting the Gaussian in G from which to sample a new task using their respective LP, this approach allows to adaptively modulate the task sampling between both, shifting the sampling towards IN when ALP-GMM does not detect high-LP subspaces and towards ALP-GMM when the current GMM of IN have lower-LP Gaussians. While combining ALP-GMM to IN, we reduce the residual random sampling of ALP-GMM from $\rho_{high} = 10\%$, used for the pretrain phase, to either $\rho_{low} = 2\%$ for experiments presented in sec. 4.1 and sec. 4.3, or $\rho_{low} = 0\%$ for experiments done in the Parkour environment in sec. 4.2 (here we found $\rho_{low} = 0\%$ to be beneficial in terms of performances w.r.t. $\rho_{low} = 2\%$, which means that the task-exploration induced by the periodic GMM fit of ALP-GMM was sufficient for exploration). In AGAIN-R and AGAIN-T, when the last GMM $p(T)$ of the IN curriculum is reached, we switch the fixed LP_{Ti} values of all IN Gaussians to periodically updated LP estimates, i.e. we allow AGAIN to modulate the importance of $p(T)$ for task sampling depending on its current student's performance.

Algorithm 2 Pretrain phase (helper function)

Require: Student policy s_θ, teacher training history $\mathcal{H}$, task-encoding parameter space $\mathcal{P}$, LP threshold δ_{LP}, experimental pre-train budget K_{pre}, pre-test set size m, number of neighbors for student selection k, random sampling ratio ρ_{high}, parametric procedural environment generator E

1: Init s_θ, train it for K_{pre} env. steps with ALP-GMM($\rho_{high}, \mathcal{P}$)
2: Pre-test s_θ with m tasks selected uniformly over $\mathcal{P}$ and get KC_s^{pre} ▷ Pre-test phase
3: Apply knn algorithm in KC space of $\mathcal{H}$, get k students closest to KC_s^{pre}
4: Among those k, keep the one with highest summed post training KC^{post}, extract its $\mathcal{C}_{raw}$
5: Get $\mathcal{C}$ from $\mathcal{C}_{raw}$ by removing any Gaussian with $LP_{ti} < \delta_{LP}$.
6: **Return** $\mathcal{C}$

Algorithm 3 Inferred progress Niches - Time-based (IN-T)

Require: Student policy s_θ, teacher training history $\mathcal{H}$, task-encoding parameter space $\mathcal{P}$, LP threshold δ_{LP}, update rate N, experimental budget K, experimental pre-train budget K_{pre}, pre-test set size m, number of neighbors for student selection k, random sampling ratio ρ_{high}, parametric procedural environment generator E

Launch Pretrain phase and get expert GMM list $\mathcal{C}$ ▷ See algo. 2
Initialize expert curriculum index i_c to 0
loop ▷ Stop after $K - K_{pre}$ environment steps
    Set i_c to $min(i_c + 1, len(\mathcal{C}))$
    Set current GMM G_{IN} to i_c^{th} GMM in $\mathcal{C}$
    loop N times
        Sample p from a Gaussian in G_{IN} chosen proportionally to its LP_{ti}
        Send $E(a \sim \mathcal{A}(p))$ to student s_θ
Add student's training trajectory to $\mathcal{H}$
Return s_θ

Algorithm 4 Inferred progress Niches - Pool-based (IN-P)

Require: Student policy s_θ, teacher training history $\mathcal{H}$, task-encoding parameter space $\mathcal{P}$, LP threshold δ_{LP}, update rate N, experimental budget K, experimental pre-train budget K_{pre}, pre-test set size m, number of neighbors for student selection k, random sampling ratio ρ_{high}, parametric procedural environment generator E

Launch Pretrain phase and get expert GMM list $\mathcal{C}$ ▷ See algo. 2
Initialize pool GMM G_{IN}, containing all Gaussians from $\mathcal{C}$
loop ▷ Stop after $K - K_{pre}$ environment steps
    Sample p from a Gaussian in G_{IN} chosen proportionally to its LP_{ti}
    Send $E(a \sim \mathcal{A}(p))$ to student s_θ
Add student's training trajectory to $\mathcal{H}$
Return s_θ

Algorithm 5 Inferred progress Niches - Reward-based (IN-R)

Require: Student policy s_θ, teacher training history $\mathcal{H}$, task-encoding parameter space $\mathcal{P}$, LP threshold δ_{LP}, update rate N, experimental budget K, experimental pre-train budget K_{pre}, pre-test set size m, number of neighbors for student selection k, random sampling ratio ρ_{high}, parametric procedural environment generator E

Launch Pretrain phase and get expert GMM list $\mathcal{C}$ ▷ See algo. 2
Initialize reward First-in-First-Out window $\mathcal{W}$, set max size to N
Initialize expert curriculum index i_c to 0
loop ▷ Stop after $K - K_{pre}$ environment steps
    If $\mathcal{W}$ is full, compute mean reward μ_w from $\mathcal{W}$
        If μ_w superior to i_c^{th} reward threshold in $\mathcal{R}$, set i_c to $min(i_c + 1, len(\mathcal{C}))$
    Set current GMM G_{IN} to i_c^{th} GMM in $\mathcal{C}$
    Sample p from a Gaussian in G_{IN} chosen proportionally to its LP_{ti}
    Send $E(a \sim \mathcal{A}(p))$ to student s_θ and add episodic reward r_p to $\mathcal{W}$
Add student's training trajectory to $\mathcal{H}$
Return s_θ

Algorithm 6 Alp-Gmm And Inferred progress Niches (AGAIN)

Require: Student policy s_θ, teacher training history $\mathcal{H}$, task-encoding parameter space $\mathcal{P}$, LP threshold δ_{LP}, update rate N, experimental budget K, experimental pre-train budget K_{pre}, pre-test set size m, number of neighbors for student selection k, random sampling ratio ρ_{low} and ρ_{high}, parametric procedural environment generator E

Launch Pretrain phase and get expert GMM list $\mathcal{C}$ ▷ See algo. 2
Setup new ALP-GMM($\rho_{rnd} = 0, \mathcal{P}$) ▷ See algo. 1
Setup either IN-T, IN-P or IN-R ▷ See algo. 3, 4 and 5
loop ▷ Stop after $K - K_{pre}$ environment steps
    Get composite GMM G from the current GMM of both ALP-GMM and IN
    $\rho_{low}\%$ of the time, sample a random parameter $p \in \mathcal{P}$
    Else, sample p from a Gaussian chosen proportionally to its LP
    Send $E(a \sim \mathcal{A}(p))$ to student s_θ and observe episodic reward r_p
    Send (p, r_p) pair to both ALP-GMM and IN
Add student's training trajectory to $\mathcal{H}$
Return s_θ

C Considered ACL and Meta-ACL teachers

Meta-ACL variants Our proposed approach, AGAIN, is based on the combination of an inferred expert curriculum with ALP-GMM, an exploratory ACL approach. In section 3 and appendix B, we present 3 approaches to use such an expert curriculum, giving the AGAIN-R, AGAIN-P and AGAIN-T algorithms. In our experiments, we also consider ablations were we only use the expert curriculum, giving the IN-R, IN-P and IN-T variants. We also consider two additional AGAIN variants that do not use our proposed KC-based student selection method:

- AGAIN with Random selection (AGAIN_RND), a lower-baseline ablation were we select the training trajectory τ from which to extract the expert curriculum randomly in history $\mathcal{H}$.
- AGAIN with Ground Truth selection (AGAIN_GT), an upper-baseline using privileged information. Instead of performing the knn algorithm in the KC space, this approach directly uses the true student distribution. For instance, in the Parkour environment, given a new student s, AGAIN_GT selects the k previously trained students from $\mathcal{H}$ that are morphologically closest to s (i.e. same embodiment type and closest limb sizes), and uses the training trajectory of the student with highest score j_s (see sec. 3).

Note that both for AGAIN_RND and AGAIN_GT, there is no need to pre-test the student, which means we can use the IN expert curriculum directly at the beginning of training rather than after a pre-training phase.

ACL conditions A first natural ACL approach to compare our AGAIN variants to is ALP-GMM, the underlying ACL algorithm in AGAIN. We also add as a lower-baseline a random curriculum teacher (Random), which samples tasks' parameters randomly over the task space.

In both the toy environment (sec. 4.1, toy env. for short) and the Parkour environment (sec. 4.2), we additionally compare to Adaptive Domain Randomization (ADR), an ACL algorithm proposed in [34], which is based on inflating a task distribution sampling from a predefined initially feasible task p_{easy} (w.r.t a given student). Each lower and upper boundaries of each dimension of the sampling distribution are modified independently with step size Δ_{step} whenever a predefined mean reward threshold r_{thr} is surpassed over a window (of size q) of tasks occasionally sampled (with probability ρ_b) at the sampling dimension boundary. More details can be found in [34]. In our experiments, as we do not assume access to expert knowledge over students sampled within the student distribution, we randomize the setting of p_{easy} uniformly over the task space in Parkour experiments and uniformly over the 4 possible student starting subspaces in toy env. experiments. Based on the hyperparameters proposed in [34] and on informal hyperparameter search, we use $[\rho_b = 0.5, r_{thr} = 1, \Delta_{step} = 0.05, q = 10]$ in toy env. experiments and $[\rho_b = 0.5, r_{thr} = 230, \Delta_{step} = 0.1, q = 20]$ in Parkour experiments.

In experiments described in sec 4.3, we compare our approaches to an oracle condition (Oracle), which is a hand-made curriculum that is very similar to IN-R, except that the list $\mathcal{C}$ is built using expert knowledge before training starts (i.e. no pre-train and pre-test phases), and all reward thresholds μ_r^i in $\mathcal{R}$ (see eq. 6) are set to 230, which is an episodic reward value often used in the literature as characterizing a default walker having a "reasonably efficient" walking gate in environments derived from the Box2D gym environment BipedalWalker [26, 15].In practice, Oracle starts proposing tasks from a Gaussian (with std of 0.05) located at the simplest subspace of the task space (ie. low stump height and high stump spacing) and then gradually moves the Gaussian towards the hardest subspaces (high stump height and low stump spacing) by small increments (50 steps overall) happening whenever the mean episodic reward of the DRL agent over the last 50 proposed tasks is superior to 230.

D Analysing Meta-ACL in a toy environment

In this section we report the full comparative experiments done in the toy environment, which includes comparisons with AGAIN-T and AGAIN-P to AGAIN-R, shown in table 1. We also provide visualizations of the KC-based curriculum priors selection process (see fig. 9) happening after the pretraining phase in AGAIN along with a visualization of the fixed set of 96 randomly drawn students used to perform the varying classroom experiments reported in sec. 4.1 (see fig. 8).

Additional comparative analysis Table 1 summarizes the post-training performances obtained by our considered Meta-ACL conditions and ACL baselines on the toy environment with only 4 possible students on a fixed set of 48 randomly drawn students. Meta-ACL conditions are given a training trajectory $\mathcal{H}$ created by training an initial classroom of 128 students. Using a Reward-based iterating scheme over the inferred expert curriculum (AGAIN-R and IN-R) outperforms the Time-based and Pool-based variants ($p < .001$). This result was expected as both these last two variants do not have flexible mechanisms to adapt to the student being trained. The pool based variants (AGAIN-P and IN-P), which discard the temporal ordering of the expert curriculum are the worst performing variants, statistically significantly inferior to both Reward-based and Time-based conditions ($p < .001$).

Condition	Regular	Random	Ground Truth
AGAIN-R	98.8 +- 4.8*	55.4 +- 32.2	99.8 +- 0.9*
IN-R	91.4 +- 3.4*	26.3 +- 41.1	92.5 +- 3.0*
AGAIN-T	84.3 +- 3.8	38.6 +- 34.1	89.0 +- 1.7*
IN-T	79.0 +- 12.0	30.3 +- 37.3	88.9 +- 1.7*
AGAIN-P	38.2 +- 7.5	9.3 +- 9.2	14.8 +- 1.2
IN-P	40.6 +- 6.4	9.2 +- 9.0	15.1 +- 1.2
ALP-GMM	84.6 +- 3.4		
ADR	14.9 +- 27.4		
Random	10.0 +- 0.8		

Table 1: **Experiments on the toy environment.** The average performance with standard deviation after 200k episodes is reported (48 seeds per conditions). For Meta-ACL variants we report results with column 1) the regular KC-based curriculum prior selection performed after 20k pre-training episodes, column 2) An ablation that performs the selection at random before training, and column 3) An oracle condition selecting before training the curriculum prior using student ground truth type. * Denotes stat. significant advantage w.r.t. ALP-GMM (Welch's t-test at $200k$ ep. with $p < 0.05$).

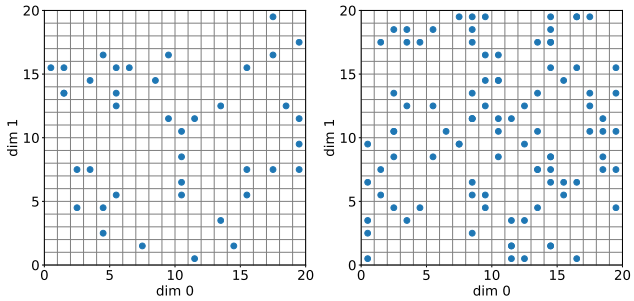
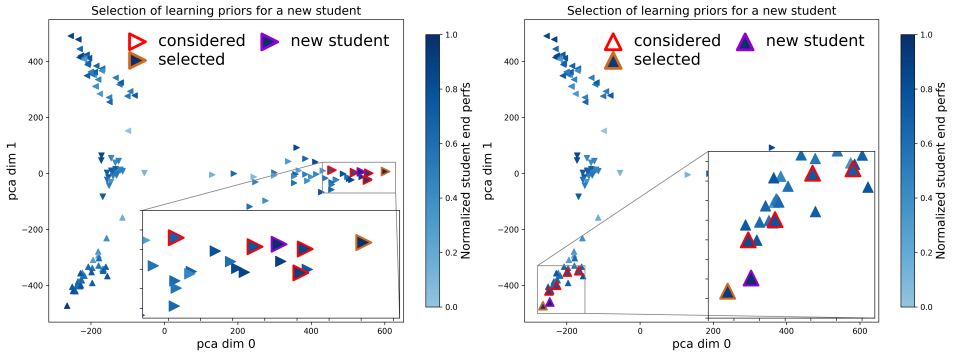
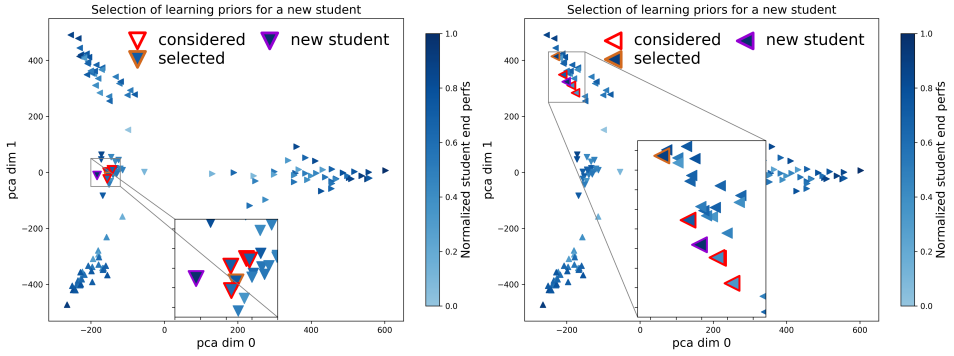


Figure 8: Additional visualizations for the varying classroom size experiments (see sec. 4.1). **Left:** Visualization of the starting cells of students from a 10% sample of a classroom of 400 students (one per student type) trained with ALP-GMM and used to populate the training trajectory $\mathcal{H}$. Each blue circle marks the starting cell of each student (i.e. its type) within the 2D parameter space $\mathcal{P}$, which is an initial learning subspace that needs to be detected by the teacher for successful training. **Right:**Visualization of the fixed set of 96 randomly drawn students that have to be trained by Meta-ACL variants given $\mathcal{H}$. As not all student types are represented in $\mathcal{H}$, Meta-ACL approaches have to generalize their curriculum generation to these new students.



(a) with type 0 new student

(b) with type 1 new student



(c) with type 2 new student

(d) with type 3 new student

Figure 9: **Examples of student selection process in 4-student type toy environment.** In all figures, we plot the 2D PCA visualization of the KC^{pre} vectors (after pre-training) of the initial classroom (128 students) trained with ALP-GMM and used to populate the training trajectory $\mathcal{H}$ used by AGAIN variants in our 4-student type toy env experiments (see sec. 4.1). We then use these 4 figures to showcase the selection process happening in 4 different AGAIN-R runs (one per student type). Each triangle represents a student, whose ground truth type (i.e. its initial learning cell) is denoted by the orientation of the triangle. Given a new student to train, AGAIN pretrains the student, constructs its KC vector (purple border triangle), infers the k closest previously trained students from $\mathcal{H}$ (red and golden border triangles), and use the one with highest end of training performance (i.e. highest score s, see sec. 3), denoted by a golden border triangle, to infer curriculum priors for the new student.

E Meta-ACL for DRL students in the Parkour environment

In this section we give additional details on the Parkour environment presented in section 4.2, and we provide additional details and visualizations on the experiments that were performed on it.

Details on the Parkour environment. In our experiments, we bound the wall spacing dimension of the task space to $\Delta_w = [0, 6]$, and the gate y position to $\mu_{gate} = [2.5, 7.5]$. In practice, given a single parameter tuple (μ_{gate}, Δ_w), we actually encode a distribution of tasks, since for each new wall along the track we add an independent Gaussian noise to each wall's gate y position μ_{gate}. Examples of parkour tasks randomly sampled within these bounds are available in figure 10 (right). At the beginning of training a given DRL policy, the agent is embodied in either a bipedal walker morphology with two joints per legs or a two-armed climber morphology with 3-joints per arms ended by a grasping "hand". Both morphologies are controlled by torque. Climbers have an additional action dimension $g \in [-1, 1]$ used to grasp: if $g \in [-1, 0[$, the climber closes its gripper, and if $g \in]0, 1]$ it keeps it open. To avoid falling (which aborts the episode with a -100 penalty) while moving forward to collect rewards, climber agents must learn to swing themselves forward by successive grasp-and-release action sequences. To increase the diversity of the student distribution, we also randomize limb sizes. See figure 10 (left) for examples of randomly sampled embodiments.

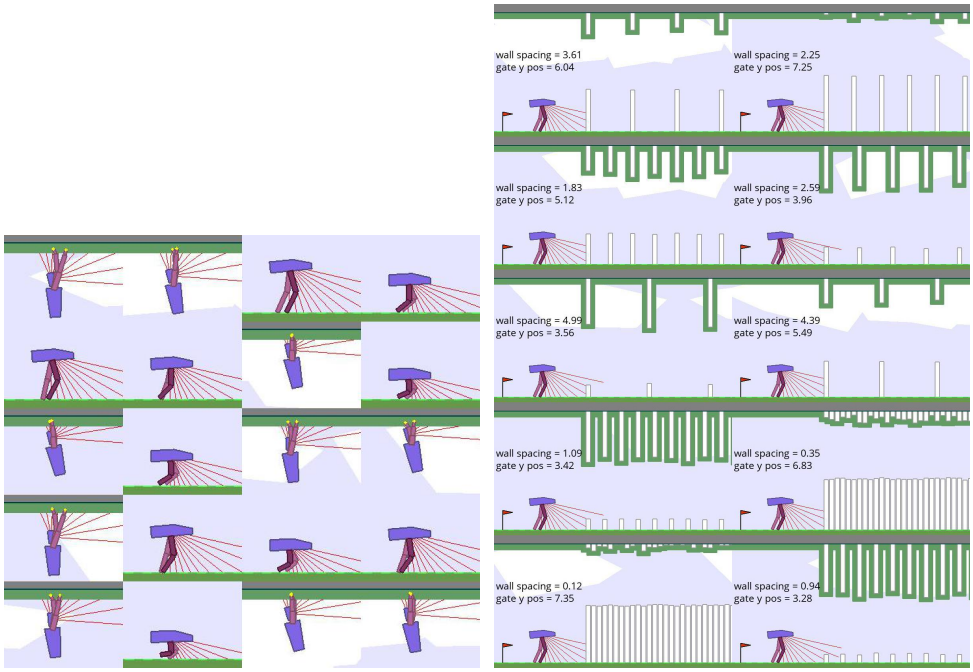


Figure 10: Visualizations of the student space and the task space of the Parkour environment.**Left:** Examples of possible agent embodiments (randomly set for a given DRL learner before training starts). **Right:** Examples of randomly sampled parkour tracks.

Soft Actor-Critic students In our experiments, we use an implementation of Soft Actor-Critic provided by OpenAI[1]. We use a 2 layered (400,300) network for V, Q1, Q2 and the policy. Gradient steps are performed each 10 environment steps, with a learning rate of 0.001 and a batch size of 1000. The entropy coefficient is set to 0.005.

Evaluation procedure To report the performance of our students on the Parkour environment, we use two separate test sets, one per embodiment type. For walkers we use a 100-tasks test set, uniformly sampled over a subspace of the task space with $\Delta_w \in [0, 6]$ and $\mu_{gate} \in [2.5, 3.6]$, which we chose based on 1) what we initially believed to be morphologically feasible for walkers, and 2) based on previously designed test sets built in recent work [15] on comparable bipedal walker experiments). For climbers, because there is no similar experiments in the literature and since it is hard to infer beforehand what will be achievable by such a morphology, we simply use a uniform test set of 225 tasks sampled over the full task space. Importantly, the

[1]https://github.com/openai/spinningup

customized test set used for walkers is solely used for visualization purposes. In our AGAIN approaches, we pre-test all students with the expert-knowledge-free set of 225 tasks uniformly sampled over the task space.

Visualizing student diversity. To assess whether our proposed multi-modal distribution of possible students in the Parkour environment do have diverse competence profiles (which is desirable as it creates a challenging Meta-ACL scenario), we plot the 2D PCA of the post training KC vector for each students of the initial classroom trained with ALP-GMM (used to populate $\mathcal{H}$). The result, visible in figure 11 (top), shows that climber-students and walker-students are located in two independant clusters, i.e. they do have clearly different competence profiles. The spread of each clusters also demonstrates that variations in initial policy parameters and limb sizes also creates students with diverse learning potentials. The competence differences between walkers and climbers can also be seen in Figure 11 (left and right), which shows the episodic reward obtained for each of the 225 tasks of the KC vector after training by a representative walker student (left) and climber student (right).

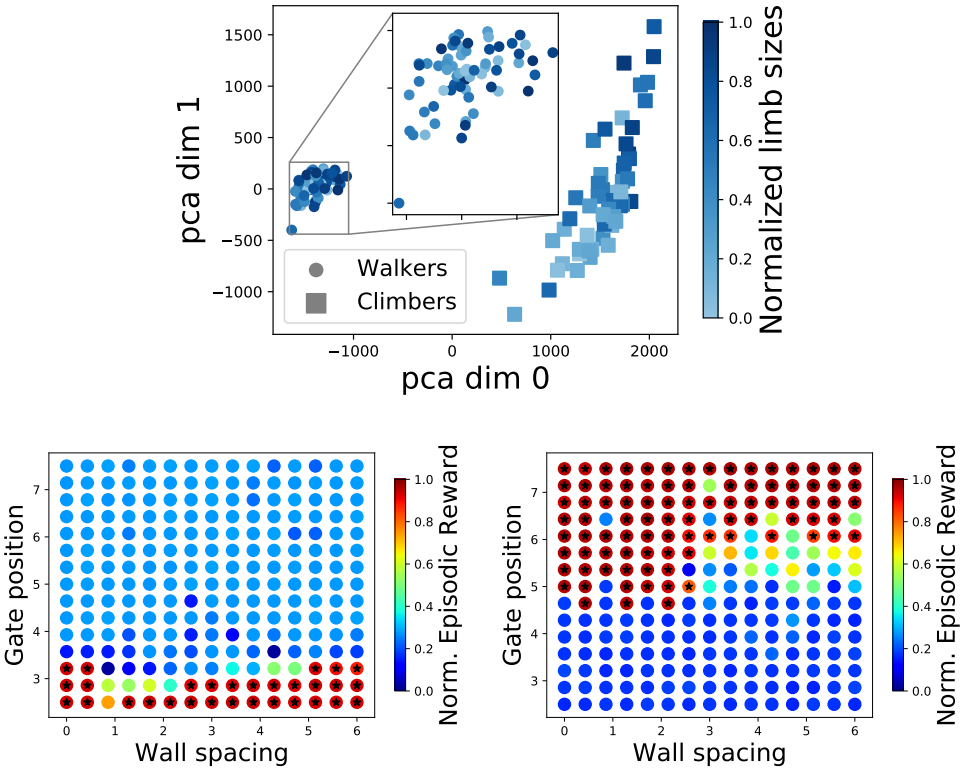


Figure 11: **top:** PCA of classroom's KC vector (128 students) after being trained for 10M student steps with ALP-GMM. **left and right:** Episodic reward obtained for each task that compose the KC vector by a walker-student (left) and a climber-student (right) of this classroom. Stars are added for all tasks for which the agent obtained more than $r = 230$ (which corresponds to an efficient locomotion policy). Walkers only manage to learn tasks with very low gate positions while climbers learn only tasks with medium to high gate positions.

F Applying Meta-ACL to a single student: Trying AGAIN instead of trying longer

In the following section we report all experiments on applying AGAIN variants to train a single DRL student (i.e. no history $\mathcal{H}$), which is briefly presented in sec. 4.3.

Parametric BipedalWalker env. We test our modified AGAIN variants along with baselines on an existing parametric BipedalWalker environment proposed in [15], which generates walking tracks paved with stumps whose height and spacing are defined by a 2D parameter vector used for the procedural generation of tasks. We keep the original bounds of this task space, i.e. we bound the stump-height dimension to $\mu_h \in [0, 3]$ and the stump-spacing dimension to $\delta_s \in [0, 6]$. As in their work, we also test our teachers when the learning agent is embodied in a modified short-legged walker, which constitutes an even more challenging scenario (as the task space is unchanged, i.e. more unfeasible tasks). The agent is rewarded for keeping its head straight and going forward and is penalized for torque usage. The episode is terminated after 1) reaching the end of the track, 2) reaching a maximal number of 2000 steps, or 3) head collision (for which the agent receives a strong penalty). See figure 12 for visualizations.

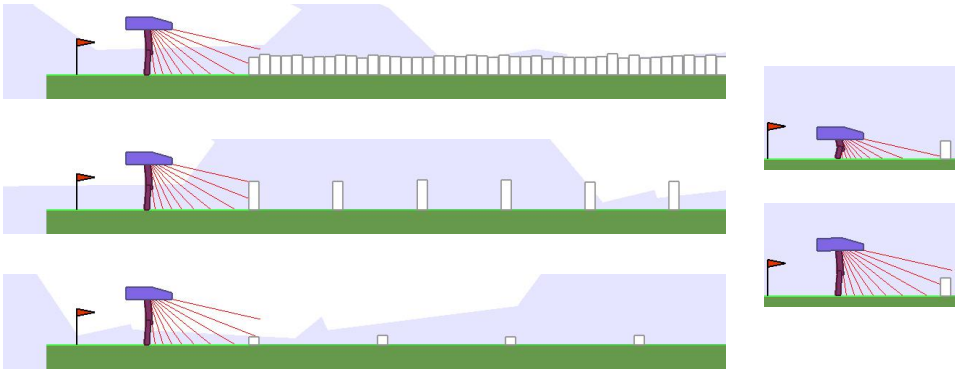


Figure 12: Parameterized BipedalWalker environment. **Left:** Examples of generated tracks. **Right:** The two walker morphologies tested on the environment. One parameter tuple (μ_h, δ_s) actually encodes a *distribution* of tasks as the height of each stump along the track is drawn from $\mathcal{N}(\mu_h, 0.1)$.

Results To perform our experiments, we ran each condition for either 10Millions (IN and AGAIN variants) or 20Millions (others) environment steps (30 repeats). The preliminary ALP-GMM runs used in IN and AGAIN variants correspond to the first 10 Million steps of the ALP-GMM condition (whose end-performance after 20 Million steps is reported in table 2. All teacher variants are tested when paired with a Soft-Actor Critic [47] student, with same hyperparameters as in the Parkour experiments (see app. E). Performance is measured by tracking the percentage of mastered tasks (i.e. $r > 230$) from a fixed test set of 100 tasks sampled uniformly over the task space. We thereafter report results for 2 independent experiments done with either default walkers or short walkers.

Is re-training from scratch beneficial? - The end performances of all tested conditions are summarized in table 2. Interestingly, retraining the DRL agent from scratch in the second run gave superior end performances than fine-tuning using the weights of the first run *in all tested variants*. This showcases the brittleness of gradient-based training and the difficulty of transfer learning. Despite this, even fine-tuned variants reached superior end-performances than classical ALP-GMM, meaning that the change in curriculum strategy in itself is already beneficial.

Is it useful to re-use ALP-GMM in the second run? - In the default walker experiments, AGAIN-R, T and P conditions mixing ALP-GMM and IN in the second run reached lower mean performances than their respective IN variants. However, the exact opposite is observed for IN-R and IN-T variants in the short walker experiments. This can be explained by the difficulty of short walker experiments for ACL approaches, leading to $16/30$ preliminary 10M steps long ALP-GMM runs to have a mean end-performance of 0, compared to $0/30$ in the default walker experiments. All these run failures led to many GMMs lists $\mathcal{C}$

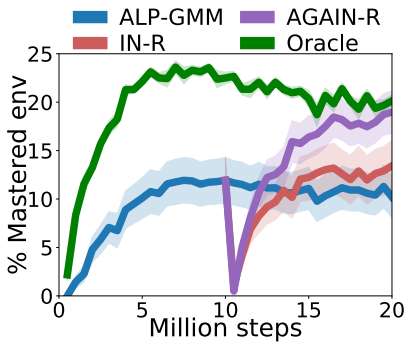


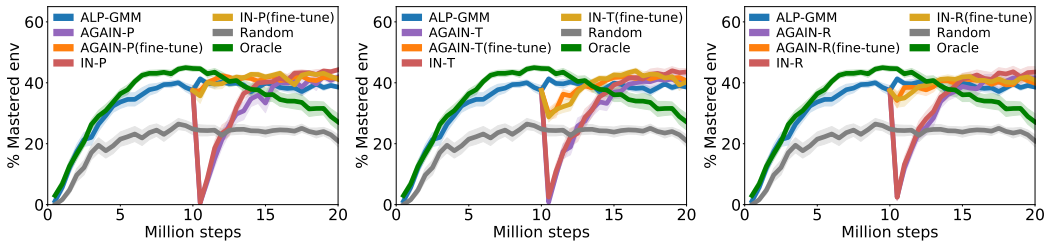
Figure 13: Given a single DRL student to train, AGAIN outperforms ALP-GMM in a parametric BipedalWalker environment. sem plotted, 30 seeds.

Condition	Short walker	Default walker
AGAIN-R	$19.0 \pm 12.0^{*}$	$41.6 \pm 6.3^{*}$
AGAIN-R(fine-tune)	11.4 ± 12.9	39.9 ± 4.6
IN-R	13.4 ± 14.4	$43.5 \pm 9.6^{*}$
IN-R(fine-tune)	11.2 ± 12.3	40.8 ± 5.6
AGAIN-T	15.1 ± 11.9	40.6 ± 11.5
AGAIN-T(fine-tune)	11.4 ± 11.8	$40.6 \pm 3.8^{*}$
IN-T	13.5 ± 13.3	$43.5 \pm 6.1^{*}$
IN-T(fine-tune)	10.7 ± 12.3	40.3 ± 7.6
AGAIN-P	13.6 ± 12.5	$41.9 \pm 5.1^{*}$
AGAIN-P(fine-tune)	11.1 ± 12.0	$41.5 \pm 3.9^{*}$
IN-P	14.5 ± 12.6	$\mathbf{44.3} \pm 3.5^{*}$
IN-P(fine-tune)	12.2 ± 12.5	$41.1 \pm 3.8^{*}$
ALP-GMM	10.2 ± 11.5	38.6 ± 3.5
Oracle	$\mathbf{20.1} \pm 3.4^{*}$	$27.2 \pm 15.2^{-}$
Random	$2.5 \pm 5.9^{-}$	$20.9 \pm 11.0^{-}$

Table 2: **Experiments on Parametric BipedalWalker with short and default bipedal walkers.** The average performance with standard deviation after 10 Millions steps (IN and AGAIN variants) or 20 Million steps (others) is reported (30 seeds per condition). For IN and AGAIN we also test variants that do not retrain the weights of the policy used in the second run *from scratch* but rather *fine-tune* them from the preliminary run.$^{*/-}$ Indicates whether performance difference with ALP-GMM is statistically significant ie. $p < 0.05$ in a post-training Welch's student t-test (* for performance advantage w.r.t ALP-GMM and $^{-}$ for performance disadvantage).

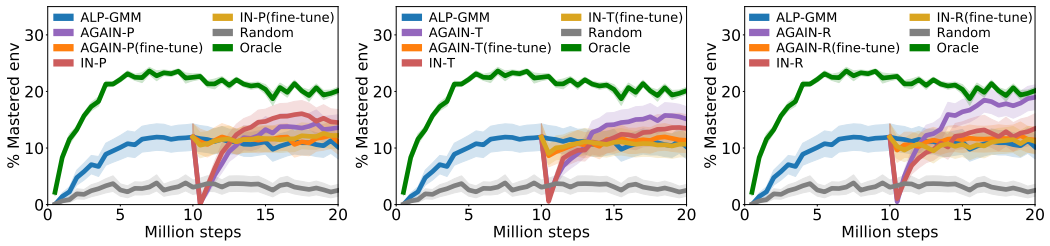
used in IN to be of very low-quality, which illustrates the advantage of AGAIN that is able to emancipate from IN using ALP-GMM.

Highest-performing variants. - Consistently with the precedent analysis, mixing ALP-GMM with IN in the second run is not essential in default walker experiments, as the best performing ACL approach is IN-P. This most likely suggests that the improved adaptability of the curriculum when using AGAIN is outbalanced by the added noise (due to the low task-exploration). However in the more complex short walker experiments, mixing ALP-GMM with IN is essential, especially for AGAIN-R, which substantially outperforms ALP-GMM and other AGAIN and IN variants (see fig. 6), reaching a mean end performance of 19.0. The difference in end-performance between AGAIN-R and Oracle, our hand-made expert using privileged information who obtained 20.1, is not statistically significant ($p = 0.6$).



(a) **Pool-based** IN (b) **Time-based** IN (c) **Reward-based** IN

Figure 14: **Evolution of performance across 20M environment steps of each condition with default bipedal walker.** Each point in each curve corresponds to the mean performance (30 seeds), defined as the percentage of mastered tracks (ie. $r > 230$) on a fixed test set. Shaded areas represent the standard error of the mean. Consistently with [15], which implements a similar approach, Oracle is prone to forgetting with default walkers due to the strong shift in task subspace (which is why it is not the best performing condition for default walker experiments.



(a) **Pool-based** IN (b) **Time-based** IN (c) **Reward-based** IN

Figure 15: **Evolution of performance across 20M environment steps of each condition with short bipedal walker.** Each point in each curve corresponds to the mean performance (30 seeds), defined as the percentage of mastered tracks (ie. $r > 230$) on a fixed test set. Shaded areas represent the standard error of the mean.